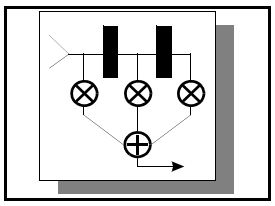
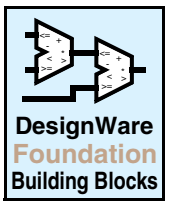




DW_fir_seq

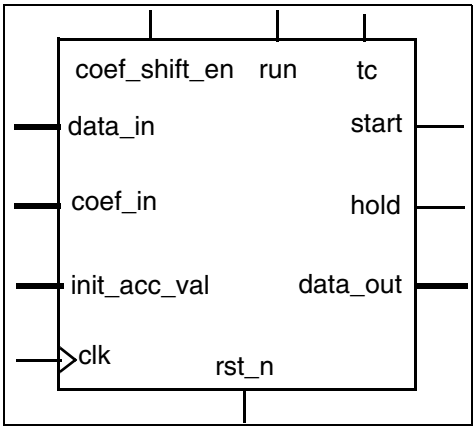
Sequential Digital FIR Filter

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Area-efficient multi-cycle implementation
- Parameterized coefficient, data, and accumulator word lengths
- Parameterized filter order
- Serially loadable coefficients
- Cascadable architecture for easy partitioning
- DesignWare datapath generator is employed for better timing and area



Applications

- 1-D FIR filtering
- Matched filtering
- Correlation
- Pulse shaping
- Adaptive filtering
- Equalization

Description

DW_fir_seq is a multi-cycle digital FIR filter processor. It is designed for Digital Signal Processing applications that employ low to medium sampling rates where area efficiency is important.

The number of coefficients in the filter as well as the coefficient, data, and accumulator word lengths are parameterized. A serial scan chain is used for loading all of the coefficients.

The processor computes the FIR convolution algorithm over multiple clock cycles in a word serial fashion with one clock cycle required per filter tap.

The processing of each data sample is synchronized with the `run` and `hold` handshake signals, which in turn are synchronous to `clk`. The `rst_n` signal is an asynchronous, active low reset that clears all coefficient and data values in the respective register files. It also resets the control sequencer and arithmetic unit to their initial states.

Table 1-1 Pin Description

Pin Name	Size	Direction	Function
clk	1 bit	Input	Clock. All internal registers are sensitive to the positive edge of `clk`.
rst_n	1 bit	Input	Asynchronous reset, active low.
coef_shift_en	1 bit	Input	Enable coefficient shift loading at `coef_in`, active high. This signal is synchronous to the positive edge of `clk`.
tc	1 bit	Input	Defines `data_in` and `coef_in` values as two's complement or unsigned. When low, the `data_in` and `coef_in` values are unsigned. When high, the `data_in` and `coef_in` values are two's complement.
run	1 bit	Input	Handshake signal that initiates the processing of a data sample on the `data_in` port. This signal is synchronous to the positive edge of `clk`.
data_in	*data_in_width* bits	Input	Input data.
coef_in	*coef_width* bits	Input	Serial coefficient load port. This port is enabled when the `coef_shift_en` pin is set high. A rising edge of `clk` loads the coefficient data at `coef_in` into the first internal coefficient register and shifts all other coefficients in the internal registers one location to the right.
init_acc_val	*data_out_width* bits	Input	Initial accumulated value for the convolution sum of products. Normally, set to zero ("000...000").
start	1 bit	Output	Handshake signal generated by synchronizing the run input with `clk`. It acknowledges the run signal and indicates the start of processing of a data_in sample.
hold	1 bit	Output	Handshake signal that indicates processing has been completed for the current data_in sample and the filter is ready to process the next sample.
data_out	*data_out_width* bits	Output	The accumulated sum of products from the FIR convolution plus the init_acc_val input $$\text{init\_acc\_val}(n-1) + \sum_{i=0}^{\text{order}-1} \text{data\_in}(n-i-1)\ \text{coef}(i)$$

Table 1-2 Parameter Description

Parameter	Values	Description
data_in_width	≥ 1	Input data word length
coef_width	≥ 1	Coefficient word length
data_out_width[a]	≥ 1	Accumulator word length
order	2 to 256	FIR filter order

a. The parameter data_out_width is normally set to a value of *coef_width + data_in_width + margin*. The value *coef_width + data_in_width* accounts for the internal coefficient multiplications. An appropriate margin must be included if the filter coefficients have a gain or are cascaded. The value *margin* ≤ log2(*order*).

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Required
str	Structural synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW03.DW_FIR_SEQ_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW_fir_seq_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_fir_seq.v	Verilog simulation model source code

Table 1-5 Modes of Operation

rst_n	coef_shift_en	Mode	Operation
1	1	Coefficient load	Serially load coefficients into the filter starting with coef(0). See Table 1-6 on page 6.
1	0	Filter	$\text{data\_out}(n) = \text{init\_acc\_val}(n-1) + \sum_{i=0}^{\text{order}-1} \text{data\_in}(n-i-1)\ \text{coef}(i)$
0	X	Reset	Asynchronously clear all internal registers to zero state

Functional Description

A block diagram of the DW_fir_seq filter is given in Figure 1-1 on page 5. The DW_fir_seq is clocked with the `clk` pin and is sensitive to the rising edge of `clk`. An asynchronous, active low reset pin, `rst_n`, clears all internal registers to the zero state.

The FIR digital filter processor consists of four main blocks: controller, sample data register file, coefficient register file and arithmetic unit. The controller is a finite state machine.

The sample data register file and the coefficient register file store data and coefficients, respectively. They are fully synchronous and are constructed out of arrays of edge-triggered D flip-flops which are clocked on the positive edge of `clk`. Address decoding is done internally using the DW01_decode unit.

The arithmetic unit is a datapath consisting primarily of a multiply-accumulator that computes the sum of products. Figure 1-2 on page 5 is a block diagram of the arithmetic unit. The multiplier-accumulator has a conditional accumulator load input on the `init_acc_val` port. The `init_acc_val` load is clocked off the rising edge of `clk` when the `run` pin is high. During normal filter operation, the accumulator is loaded with a zero word through `init_acc_val` on the first cycle and one filter tap is computed per cycle for `order` cycles. If a signal $x(n)$ is to be added to the filter output, you can save an adder by loading this signal into the accumulator through `init_acc_val` instead of loading zero.

The filter is programmed by serially loading coefficients into the device through the `coef_in` port when `coef_shift_en` is high. The loading sequence is clocked off the rising edge of `clk`. In this loading mode, coefficients are loaded serially into the coefficient register file, starting at `coef(0)` progressing up to `coef(order - 1)`.

During execution mode, the `run` handshake signal initiates the processing cycle for each new data sample. The computation of the FIR convolution takes one cycle per tap plus one additional cycle to empty the pipeline. Therefore, `order+1` cycles are required to complete the computation. The `hold` signal is asserted high when the computation for a sample is complete, indicating that the result is ready on the `data_out` port and the filter is ready to process the next sample.

The `tc` pin identifies the type of data entering the `data_in` port and the type of coefficients with which the data is convolved. The data and coefficient types must be the same. When `tc` is high, the data and coefficient type is two's complement. When `tc` is low, the type is unsigned.

Figure 1-1 Block Diagram

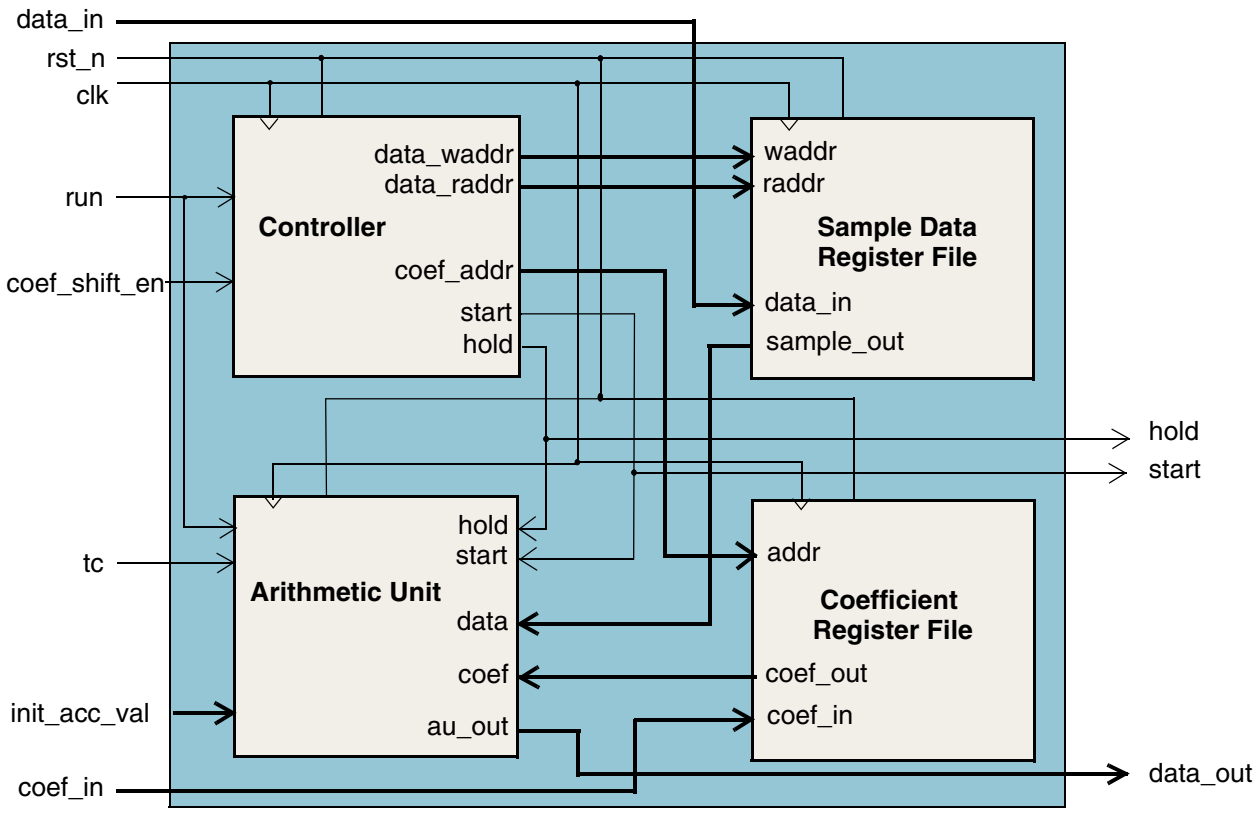


Figure 1-2 Arithmetic Unit Block Diagram

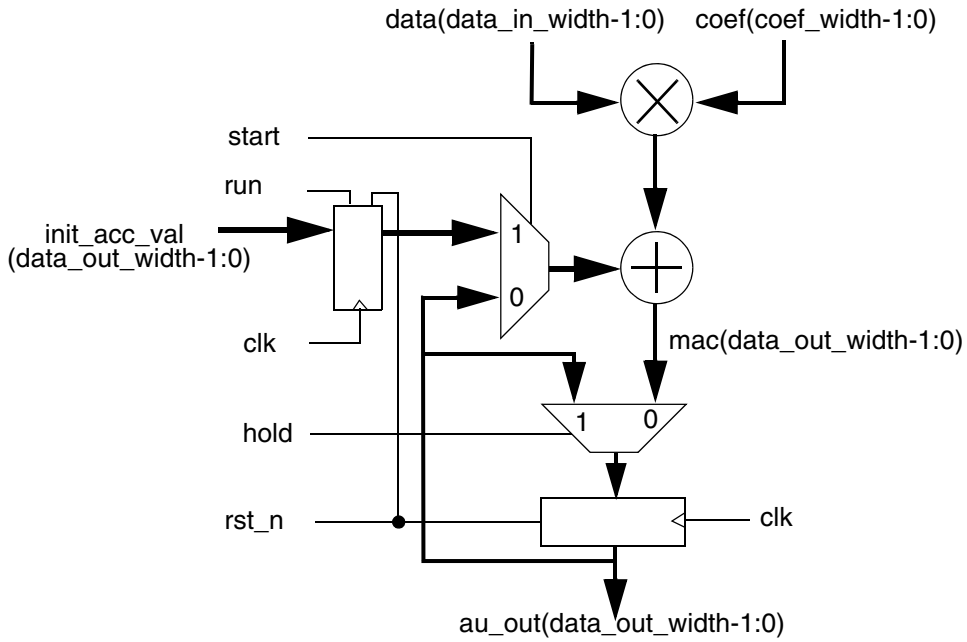


Table 1-6 Coefficient Register Loading Sequence for 8-Tap FIR Filter (order = 8)

Clock Cycle	Mode of Operation	Internal Coefficient Register State							
		7	6	5	4	3	2	1	0
0	Reset	0	0	0	0	0	0	0	0
1	Coefficient Load	coef(0)	0	0	0	0	0	0	0
2	Coefficient Load	coef(1)	coef(0)	0	0	0	0	0	0
3	Coefficient Load	coef(2)	coef(1)	coef(0)	0	0	0	0	0
4	Coefficient Load	coef(3)	coef(2)	coef(1)	coef(0)	0	0	0	0
5	Coefficient Load	coef(4)	coef(3)	coef(2)	coef(1)	coef(0)	0	0	0
6	Coefficient Load	coef(5)	coef(4)	coef(3)	coef(2)	coef(1)	coef(0)	0	0
7	Coefficient Load	coef(6)	coef(5)	coef(4)	coef(3)	coef(2)	coef(1)	coef(0)	0
8	Coefficient Load	coef(7)	coef(6)	coef(5)	coef(4)	coef(3)	coef(2)	coef(1)	coef(0)
9	Filter	coef(7)	coef(6)	coef(5)	coef(4)	coef(3)	coef(2)	coef(1)	coef(0)

Suppressing Warning Messages During Verilog Simulation

The Verilog simulation model includes macros that allow you to suppress warning messages during simulation.

To suppress all warning messages for all DWBB components, define the DW_SUPPRESS_WARN macro in either of the following ways:

- Specify the Verilog preprocessing macro in Verilog code:

  ```
  `define DW_SUPPRESS_WARN
  ```

- Or, include a command line option to the simulator, such as:

  `+define+DW_SUPPRESS_WARN` (which is used for the Synopsys VCS simulator)

The warning messages for this model include the following:

- If values other than 1 or 0 are present on a clock port, the following message is displayed:

  ```
  WARNING: <instance_path>.<clock_name>_monitor:
      at time = <timestamp>, Detected unknown value, x, on <clock_name> input.
  ```

  To suppress only this warning message for all DWBB components, use the following macro:

  - Define the DW_DISABLE_CLK_MONITOR macro. You can define this macro in the following ways:
    - Specify the Verilog preprocessing macro in Verilog code:

      ```
      `define DW_DISABLE_CLK_MONITOR
      ```

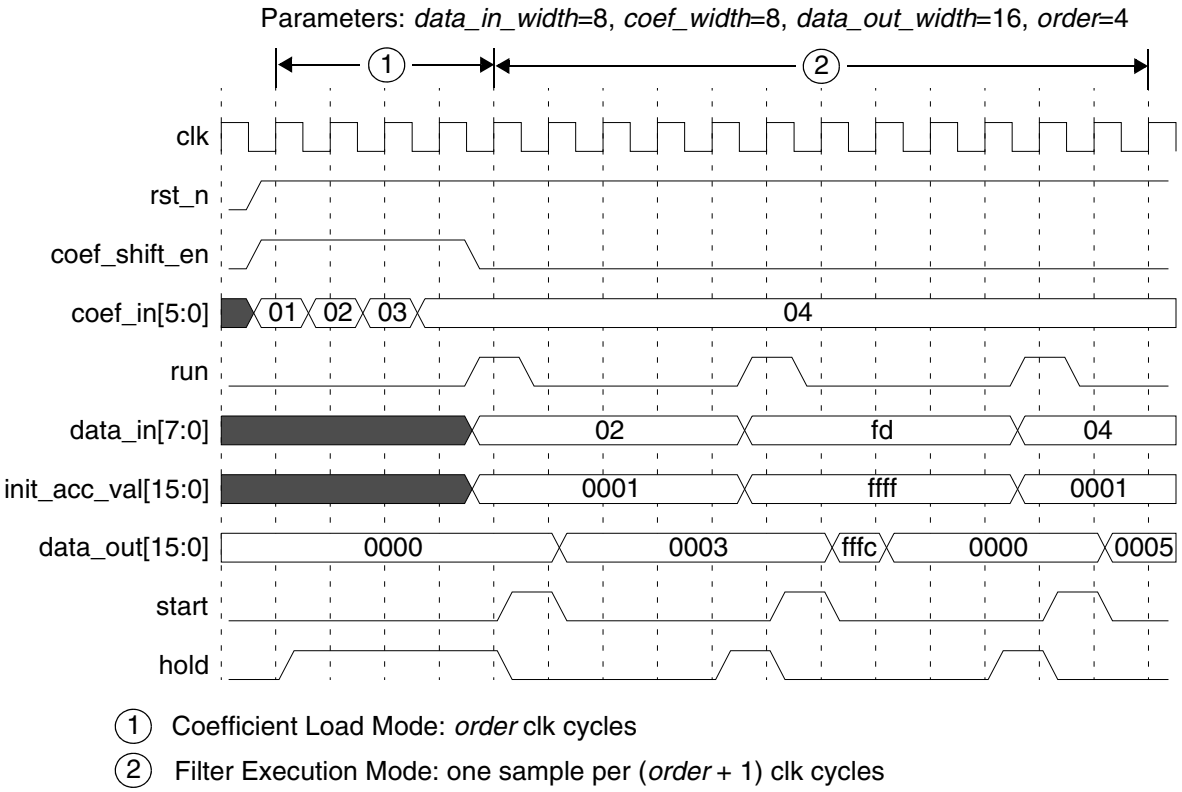
    - Or, include a command line option to the simulator, such as:

      `+define+DW_DISABLE_CLK_MONITOR` (which is used for the Synopsys VCS simulator)

  This message is also suppressed using the DW_SUPPRESS_WARN macro explained earlier.

Timing Waveforms

Figure 1-3 DW_fir_seq Timing Diagram



(1) Coefficient Load Mode: *order* clk cycles

(2) Filter Execution Mode: one sample per (*order* + 1) clk cycles

Application Example

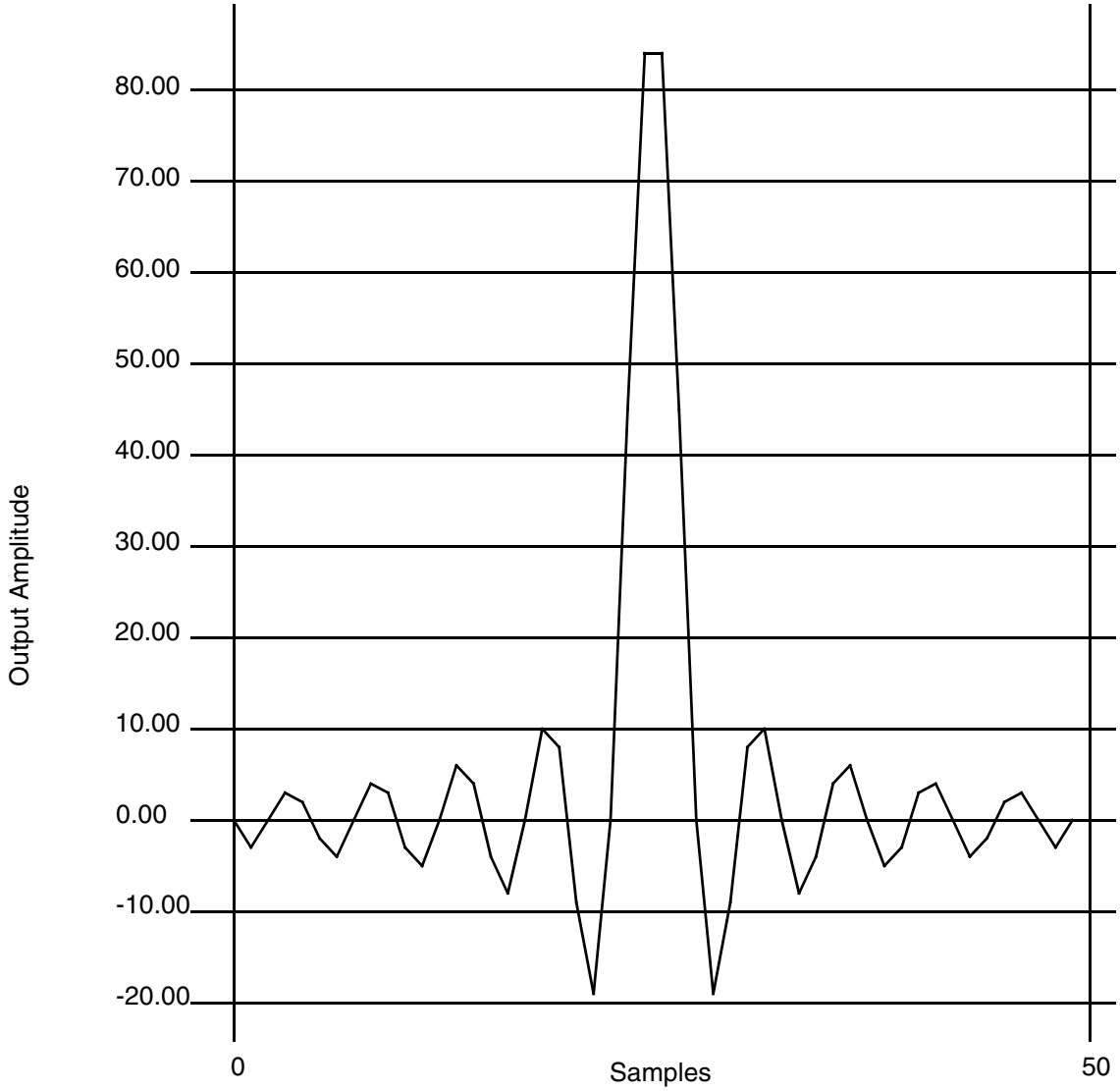
The "DW_coef Package" on page 9 defines the values required to implement a 48th order low-pass Kaiser Window filter. The coefficient values defining the Kaiser Window response are specified as a constant array. Because of the symmetry of coefficient values, only half of the values are specified. The filter impulse response is shown in Figure 1-4 on page 10.

DW_coef Package

```
-- The following VHDL code defines the package that specifies
-- the value of each parameter and the coefficient values
-- defining the Kaiser Window response.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

package DW_coef is
  -- kaiser window LP FIR Filter
  -- only half of the coefficient array is specified because of symmetricity
  constant kaiser_order: INTEGER := 48;
  constant kaiser_coef_width: INTEGER := 12;
  type coef_half_array is array (0 to kaiser_order/2-1) of
                          std_logic_vector_(kaiser_coef_width-1 downto 0);
  constant kaiser_coef_half: coef_half_array :=
    ("111111111101",
     "000000000000",
     "000000000011",
     "000000000010",
     "111111111110",
     "111111111100",
     "000000000000",
     "000000000100",
     "000000000011",
     "111111111101",
     "111111111011",
     "000000000000",
     "000000000110",
     "000000000100",
     "111111111100",
     "111111111000",
     "000000000000",
     "000000001010",
     "000000001000",
     "111111110111",
     "111111101101",
     "000000000000",
     "000000101101",
     "000001010100"
     );
end DW_coef;
```

Figure 1-4 FIR Filter Impulse Response (order = 48, Kaiser Window)



Related Topics

- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp.all;

entity DW_fir_seq_inst is
  generic (inst_data_in_width  : POSITIVE := 8;
           inst_coef_width     : POSITIVE := 8;
           inst_data_out_width : POSITIVE := 18;
           inst_order          : POSITIVE := 6 );
  port (inst_clk           : in std_logic;
        inst_rst_n         : in std_logic;
        inst_coef_shift_en : in std_logic;
        inst_tc            : in std_logic;
        inst_run           : in std_logic;
        inst_data_in : in std_logic_vector(inst_data_in_width-1 downto 0);
        inst_coef_in : in std_logic_vector(inst_coef_width-1 downto 0);
       inst_init_acc_val : in
                           std_logic_vector(inst_data_out_width-1 downto 0);
        start_inst          : out std_logic;
        hold_inst           : out std_logic;
        data_out_inst : out std_logic_vector(inst_data_out_width-1 downto 0)
        );
    end DW_fir_seq_inst;

architecture inst of DW_fir_seq_inst is
begin
  -- Instance of DW_fir_seq
  U1 : DW_fir_seq
    generic map ( data_in_width => inst_data_in_width,
                  coef_width => inst_coef_width,
                  data_out_width => inst_data_out_width,
                  order => inst_order )
    port map ( clk => inst_clk,   rst_n => inst_rst_n,
               coef_shift_en => inst_coef_shift_en,   tc => inst_tc,
               run => inst_run,   data_in => inst_data_in,
               coef_in => inst_coef_in,   init_acc_val => inst_init_acc_val,
               start => start_inst,   hold => hold_inst,
               data_out => data_out_inst );
end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_fir_seq_inst(inst_clk, inst_rst_n, inst_coef_shift_en, inst_tc,
                       inst_run, inst_data_in, inst_coef_in,
                       inst_init_acc_val, start_inst, hold_inst,
                       data_out_inst );

  parameter data_in_width = 8;
  parameter coef_width = 8;
  parameter data_out_width = 18;
  parameter order = 6;

  input inst_clk;
  input inst_rst_n;
  input inst_coef_shift_en;
  input inst_tc;
  input inst_run;
  input [data_in_width-1 : 0] inst_data_in;
  input [coef_width-1 : 0] inst_coef_in;
  input [data_out_width-1 : 0] inst_init_acc_val;
  output start_inst;
  output hold_inst;
  output [data_out_width-1 : 0] data_out_inst;

  // Instance of DW_fir_seq
  DW_fir_seq #(data_in_width, coef_width, data_out_width, order)
    U1 ( .clk(inst_clk),   .rst_n(inst_rst_n),
         .coef_shift_en(inst_coef_shift_en),   .tc(inst_tc),
         .run(inst_run),   .data_in(inst_data_in),   .coef_in(inst_coef_in),
         .init_acc_val(inst_init_acc_val),   .start(start_inst),
         .hold(hold_inst),   .data_out(data_out_inst) );
endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
July 2020	DWBB_201912.5	■ Adjusted content and title of "Suppressing Warning Messages During Verilog Simulation" on page 7 and added the DW_SUPPRESS_WARN macro
October 2019	DWBB_201903.5	■ Added the "Disabling Clock Monitor Messages" section ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com